Computer Vision II - Homework Assignment 5
BONUS

Stefan Roth, Anne Wannenwetsch, Stephan Richter, TU Darmstadt

July 14, 2017

This *bonus* homework is due on August 8, 2015 at 09:00.

Goals.

This assignment will deepen your understanding of inference on Markov random fields using graph cuts.

Problem 1 - Iterated Graph Cuts - 26 points

As an application of graph cuts, you are going to implement a simplified version of the GrabCut algorithm [RKB04] for figure-background segmentation. The inputs for this algorithm are an image as well as a bounding box annotation drawn around the object. The output is a binary label for each pixel either classifying it as foreground or background. Pixels outside the bounding box are fixed to be in the background. With I, Θ and S denoting the image, the parameters of the color distribution and the segmentation, respectively, the segmentation energy for pixels inside the bounding box is given by

$$E(S \mid I, \Theta) = E_d(S \mid I, \Theta) + \lambda E_s(S \mid I) \tag{1}$$

where E_d and E_s are the data term and the smoothness term, respectively. The data term

$$E_d(S \mid I, \Theta) = \sum_k -\log \mathrm{GMM}(I_k \mid \theta_{S_k}). \tag{2}$$

measures how well each pixel fits to the color distribution of foreground or background pixels (depending on the segmentation). Color distributions are represented as Gaussian Mixture Models with 5 components.

The spatial smoothness term favors segmentations that align to image boundaries by using contrast sensitive Pott's potentials on a 4-connected MRF:

$$E_s(S \mid I) = \sum_{(i,j) \in \mathcal{N}} \exp(-\beta \|I_i - I_j\|_2^2) \cdot \delta_{S_i \neq S_j} \tag{3}$$

The version you are going to implement proceeds by initializing the color distributions from user annotation (e. g. a bounding box) and then iterating between minimizing the segmentation energy and re-fitting the color distributions. In the original GrabCut paper, a joint energy over S and Θ is defined, but we will ignore this for simplicity.

Tasks:

- Implement the function

  ```
  E = make_edges(height, width),
  ```

  which creates edges for a four-connected grid graph, *i.e.* a graph where each node is connected to its four neighbors. The result should be an $|E| \times 2$ matrix, where $|E|$ is the total number of edges. Each row indicates the linear index of the two nodes which are connected by that edge.

  3 points

- Implement the function

  ```
  G,E,s,t = make_graph(height,width),
  ```

  which calls `make_edges` and creates the graph G with source s and sink t in order to solve a maxflow/mincut problem.

  4 points

- Implement the function

  ```
  W = contrast_weight(img, E),
  ```

  which computes the weight term for the contrast sensitive Pott's potential. Set the missing parameter β as follows:

  $$\beta = \left(\frac{1}{|\mathrm{E}|} \sum_{(i,j) \in \mathrm{E}} \|I_i - I_j\|_2^2 \right)^{-1}$$

  `W` should be a column vector with the same number of rows as `E`.

  3 points

- Implement the function

  ```
  S = smoothness_term(E, W, lambda, n),
  ```

  which assembles the edge information, the accompanying weights and the smoothness factor λ into a sparse $n \times n$ matrix. For each (i, j) in E the matrix S holds the effective weight $S_{i,j}$ for the Pott's potential at this edge.

  2 points

- Implement the function

  ```
  [fg_gmm, bg_gmm] = fit_colors(img, mask, k),
  ```

  which estimates Gaussian mixture models `fg_gmm` and `bg_gmm` for foreground and background pixels, respectively. The `GaussianMixtures` package might be of help here to approximately estimate the maximum likelihood of the parameters. Here, `img` denotes a RGB color image that is segmented with the `mask` into foreground (`true`) and background (`false`). The number of mixture components is passed by the variable k.

  3 points

- Implement the function

```
D = data_term(img, fg_gmm, bg_gmm, s, t),
```

which returns a sparse matrix holding the data term for each pixel. s and t are the indices for source and sink nodes.

3 points

- Implement the function

  ```
  seg = iterated_graphcuts(img, bbox, lambda, k),
  ```

  which puts together the previously implemented components into the final segmentation algorithm. Here, `img` denotes an RGB color image on which a bounding box `bbox = [top, height, left, width]` is annotated. Initialize all pixels within the bounding box as foreground and do 10 iterations of graph cuts and color distribution estimation. The final output should be the segmented image `seg`. Display the current foreground estimate after each iteration. You might find the algorithm of *Boykov & Kolmogorov* in the `LightGraphs` package helpful for solving the maxflow/mincut problem.

  5 points

- Run the script `problem1` that calls your iterated graph cuts algorithm with a test image and predefined bounding box and shows the result. Comment on the quality of your result. If you see artifacts, why might they be there? What could be done to improve the results?

  3 points

Notes:

- In order to foster modularity, each function is thought to use as little prior information about the parameters as possible. Hence all functions but `iterated_graphcut` do not know about the bounding box.
- The color distributions should be fitted by using all pixels, i. e. the pixels inside as well as outside the boundary.
- When implementing Graphcuts in MATLAB, we would use the original GCO package with a MEX interface and suppporting sparse matrices. As the so far only Julia port of this package seems to be less mature, we fallback to solving a maxflow/mincut problem directly in Julia using the `LightGraphs` package. This however lacks support for sparse matrices. If you hence run out of memory during your computations, you are free to use our lab computers and/or scale down the input image.
- You *must* adhere to the function signatures and semantics of inputs/outputs. Put up a question on Moodle if you have any doubts.

References

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “GrabCut”: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH ’04, pages 309–314, New York, NY, USA, 2004. ACM.